

Lawrence Berkeley National Laboratory

Lawrence Berkeley National Laboratory

Title

Exploration of cache behavior using HPSS per-file transfer logs

Permalink

<https://escholarship.org/uc/item/7c8155dt>

Author

Holmes, Harvard

Publication Date

2001-11-10

Exploration of Cache Behavior Using HPSS Per-File Transfer Logs

Harvard Holmes (hholmes@lbl.gov)

Mass Storage Group
Nancy Meyer (Group Lead),
Matthew Andrews, Shreyas Cholia, Damian Hazen,
Wayne Hurlbert, Nancy Johnston

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

November 2001

Abstract

We assembled 18 months of transfer logs from a production High Performance Storage System (HPSS) system at the National Energy Research Scientific Computing Center (NERSC) and analyzed them to assess workload behavior and gain some insight into which cache configurations would provide the best service to the users.

We found, as expected, that the workload is distributed over file size with a declining number of files as the files get larger, so the amount of space consumed per file size increment is roughly constant up to file sizes of 1 GB. Sixty one percent of file accesses were write accesses. There are a significant number of files written which are never read – backup files and similar files. For all sizes of files, access frequencies decline with the age of the files.

HPSS uses the cache as an I/O buffer for incoming data. At our installation the cache behavior is dominated by the write traffic. Cache lifetimes tend to scale linearly with the size of the cache and inversely with the amount of data flow.

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Introduction

HPSS was developed by a consortium of government, industrial, and educational sites [1] and is currently deployed at many large supercomputing sites. NERSC is a developer site within the consortium. HPSS provides a high performance data storage architecture based on the IEEE Mass Storage Reference Model [3, 4], which is designed to operate in a multiple-computer parallel configuration. The HPSS facility is a key component among the computational resources provided to NERSC users, responding to the increasingly data-intensive aspects of modern computational science.

NERSC has two HPSS systems ("Archive" and "HPSS"), both of which are accessible interactively from all NERSC supercomputers and auxiliary systems, as well as from off-site computers. Both systems are also accessible to batch jobs from all NERSC systems [2]. Of the two systems, user archive activity is encouraged on "Archive," and backup and other system activities are encouraged on "HPSS." Each system has an IBM AIX SP Silver node for file system management and other metadata operations, and 4 Winterhawk nodes for data movement. The systems had 3 TB (Archive) and 3.3 TB (HPSS) of disk cache at the time of this study. The systems share 60 StorageTek 9840 tape drives in 8 StorageTek tape libraries. Collectively, the systems store about 200 TB of data in about 12 million files, and perform about 900 GB of user I/O per day [5]

NERSC 's HPSS Hardware Architecture

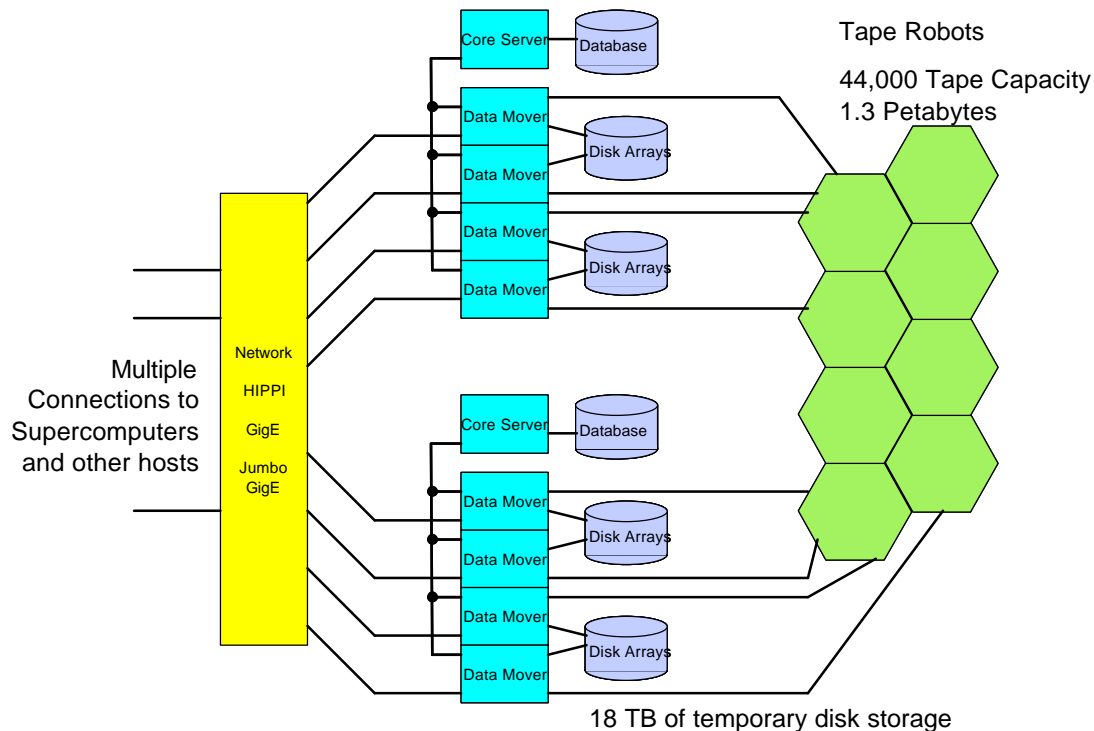


Figure 1. NERSC's Hardware Architecture

Configuration Choices in HPSS

Both systems are configured with multiple Classes of Service (COS), which segment the disk cache according to file sizes and other properties. Users may also specifically select a COS for special file usages, such as backups. Generally the systems have COSs for small (0 to 2 MB) files, medium (2 MB to 100 MB) files, and large (over 100 MB) files. Some sites also have categories of service similar to the above which provide multiple tape copies of files for offsite storage.

More detailed configuration choices in HPSS involve the allocation of disk resources to specific storage classes, the configuration of disk resources into allocation blocks, and the choice of purge and migration policies.

Issues Involved in Selecting Storage Class Sizes and COS Size Limits

- Minimize wasted space on disk and tape;
- Maximize writing efficiency to tape (taken as constant in this analysis);
- Minimize reading costs;
- Minimize repack costs (not addressed in this analysis).

Data Collection

The primary interfaces to our HPSS systems are PFTP and HSI. PFTP is the HPSS supplied parallel FTP client, and HSI is the Hierarchical Storage Interface developed by Mike Gleicher with funding from several of the development sites. Both of these interfaces create both detailed and transfer log files as text files. The detail logs record a great deal of internal operational data, while the transfer logs record one entry for each file transferred to or from a client. The transfer logs are saved and used for statistical summaries and to extract individual usage information. Each interface type creates its own distinct log file on the machine where it runs and the log files of different types and from the various machines must be combined later. The transfer logs were deemed most suitable for this investigation. Information included in the transfer logs includes the date and time of the transfer, the time taken for the transfer or the data rate, the host which was the client, the size and name of the file transferred, and the user who transferred the file (not necessarily the owner of the file). Unfortunately, the transfer logs do not include information about file deletions. The lack of file deletion information prevents the complete reconstruction of the state of the cache, but the overall growth of the system is not largely different from the growth indicated by the transfer logs, so we believe that our exploration of cache behavior is still relevant. It is also the case that many of the files which are deleted are backup files, which are not deleted until weeks or months after they are stored. These files have generally already been purged from the cache, so their lack of deletion information in the log files does not affect our analysis.

Over time, the formats of the transfer records have had several changes and revisions. A PERL script was written to examine each record and deduce the format and rewrite selected information in a standard form, including:

- the date and time of the transfer,
- the size of the transfer,
- whether the transfer was a read or write, and
- the full path name of the transfer (occasionally only the name is available).

This standard form is then sorted by the full path name. Workload characterization is performed using this sorted file. Using the sorted file, we replaced file path names with sequence numbers to make the file smaller for subsequent cache simulations. However, we added a field for the file extension, in case we wished to categorize files by type of file in later analyses. This created a second standard form, similar to the first form, but with the added field of the file extension. This second form was then sorted by date and time and passed to the cache simulation routines.

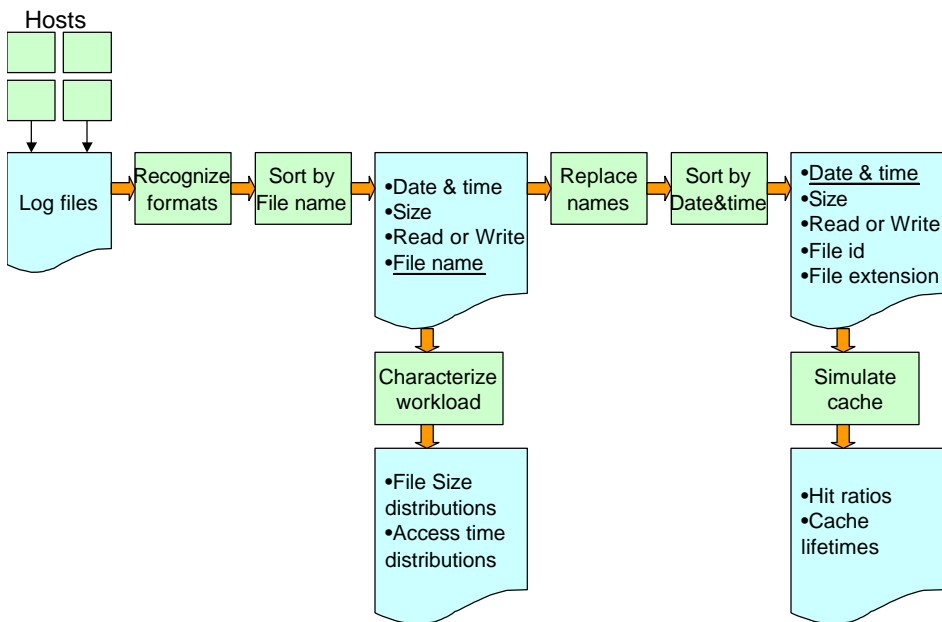


Figure 2. Data Processing Flow

Workload Characterization

During the time of the trace files, the total space utilized in both storage systems at NERSC grew from 111 TB at the end of 1999 to 213 TB in July of 2001 for a growth of 102 TB. Our logs show the total number of files written or overwritten as 178 TB. Subtracting this number from the growth yields the estimate that 76 TB of files were deleted or overwritten during the period.

File Counts by Size Category

Our first workload characteristic is simply the file counts and the amount of space transferred. The counts and amounts are listed for the totals and for the read subset. The

"newfiles" are the number of unique file names in the log files. The analysis program does not have access to the existing directory of files, so a "newfile" can arise through writing a truly new file or through reading a file that has not been seen before by the analysis program. The graph of space density is obtained by dividing the GB in each category by the width of the category ("upto_GB" minus "from GB").

```

#=====
# "Archive"
# Summary of io activity from transfer logs
#=====
# from_GB upto_GB io_GB read_GB io_cnt read_cnt newfiles
0.0000 0.0001 13.034 4.021 908629 235495 536329
0.0001 0.0003 38.909 10.247 225874 55551 162341
0.0003 0.0010 127.432 38.665 206953 62177 115188
0.0010 0.0030 511.700 254.441 336471 181670 147465
0.0030 0.0100 1616.117 435.771 254880 68606 191410
0.0100 0.0320 2923.993 1086.314 147252 55298 94972
0.0320 0.1000 10069.235 2951.907 183206 53459 96499
0.1000 0.3200 27894.638 17094.732 147680 89297 71808
0.3200 1.0000 37166.863 18562.653 73561 39145 40023
1.0000 3.0000 8193.525 1041.820 5639 598 4897
3.0000 10.0000 10800.173 3344.556 2616 934 1625
10.0000 32.0000 3381.402 108.978 161 6 143
32.0000 100.0000 424.257 107.627 7 2 4
100.0000 320.0000 622.263 0.000 4 0 4
320.0000 1024.0000 0.000 0.000 0 0 0

0.0000 1024.0000 103783.542 45041.732 2492933 842238 1462708

```

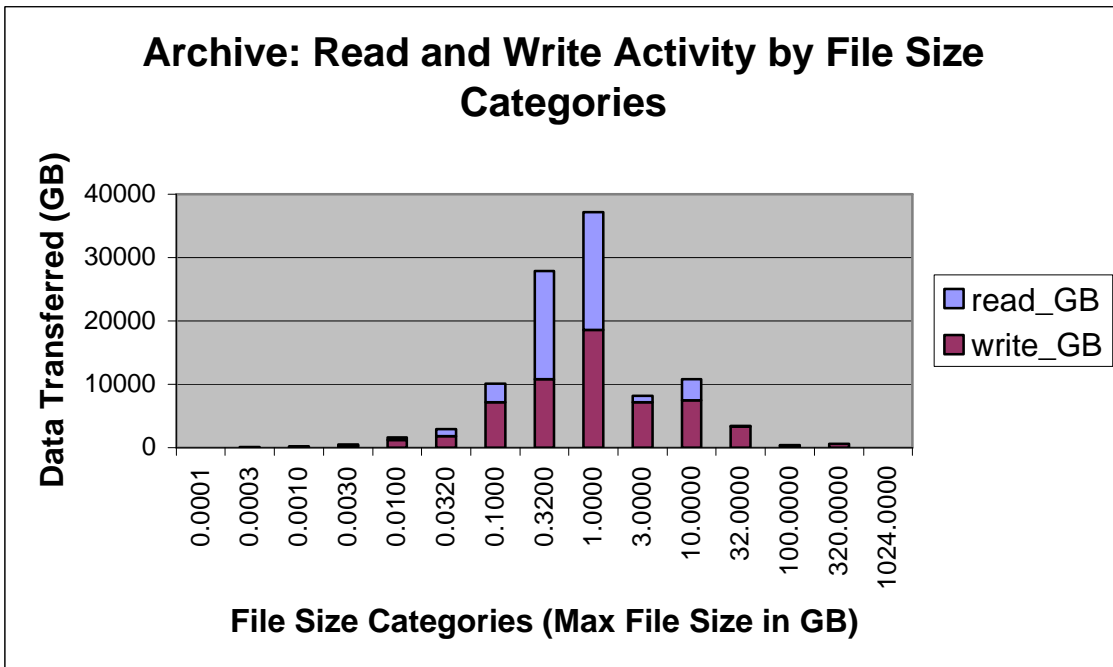


Figure 3. Read and Write Activity by File Size Categories for the Archive System

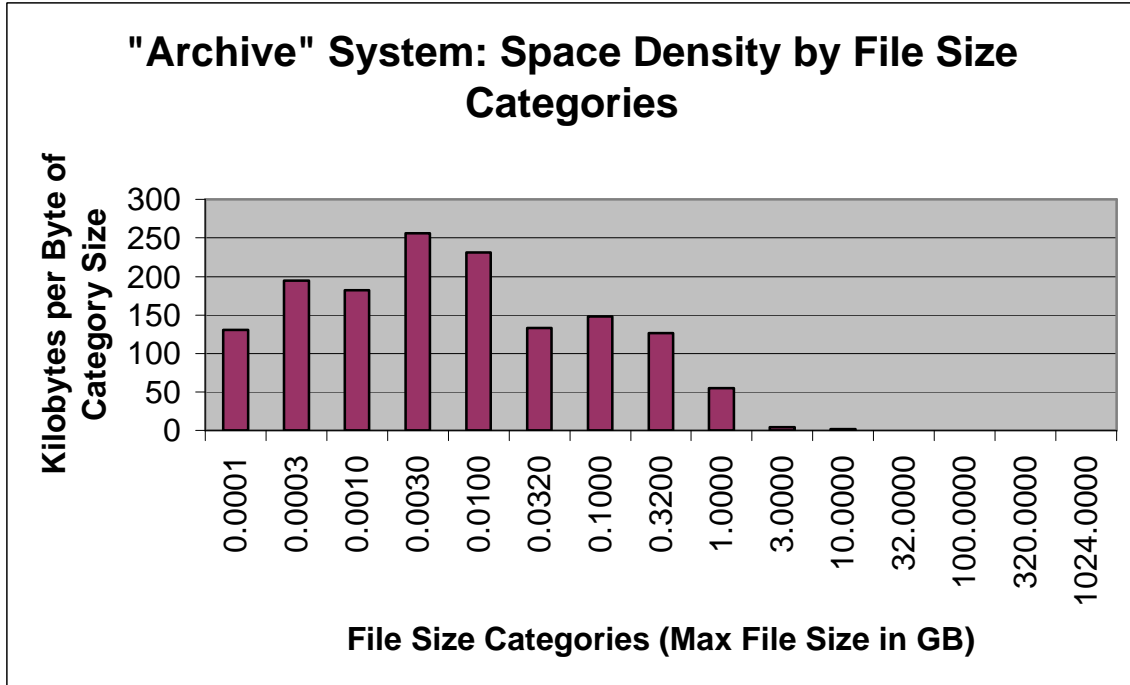


Figure 4. Space Density by File Size Categories for the Archive System

```

=====
# "HPSS"
# Summary of io activity from transfer logs
=====
# from_GB  upto_GB  io_GB  read_GB  io_cnt  read_cnt  newfiles
0.0000    0.0001    19.928  3.465   1117654  301651   600912
0.0001    0.0003    35.571  11.495  186682   60385   137826
0.0003    0.0010   208.649  53.763  353502   89654   235130
0.0010    0.0030  1222.448  804.776  802683  553072  332135
0.0030    0.0100  2745.203 1334.363  461461  223946  299510
0.0100    0.0320  6122.866 1637.871  381031  103705  263734
0.0320    0.1000 10148.328 2694.911  185211   42475  141886
0.1000    0.3200 13990.182 4555.113   74218   22306   44653
0.3200    1.0000 38262.774 7204.535   75058   13422   50445
1.0000    3.0000 20061.262 1679.059   12212    1092   10249
3.0000   10.0000 24711.444 4937.239    4755     921    3731
10.0000  32.0000 23865.494 3402.440    1550     241    1254
32.0000 100.0000  4943.268   0.000     118       0     117
100.0000 320.0000  1644.745   0.000      11       0      11
320.0000 1024.0000   0.000   0.000       0       0       0

0.0000 1024.0000 147982.160 28319.031 3656146 1412870 2121593

```

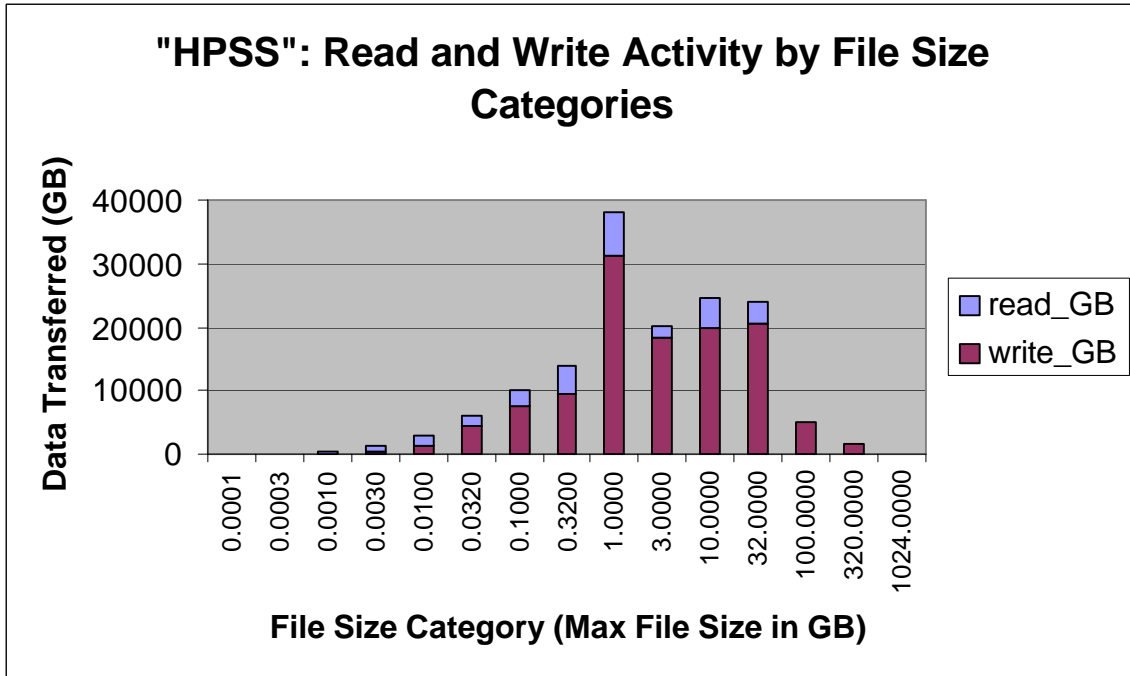


Figure 5. Read and Write Activity by File Size Categories for the "HPSS" System

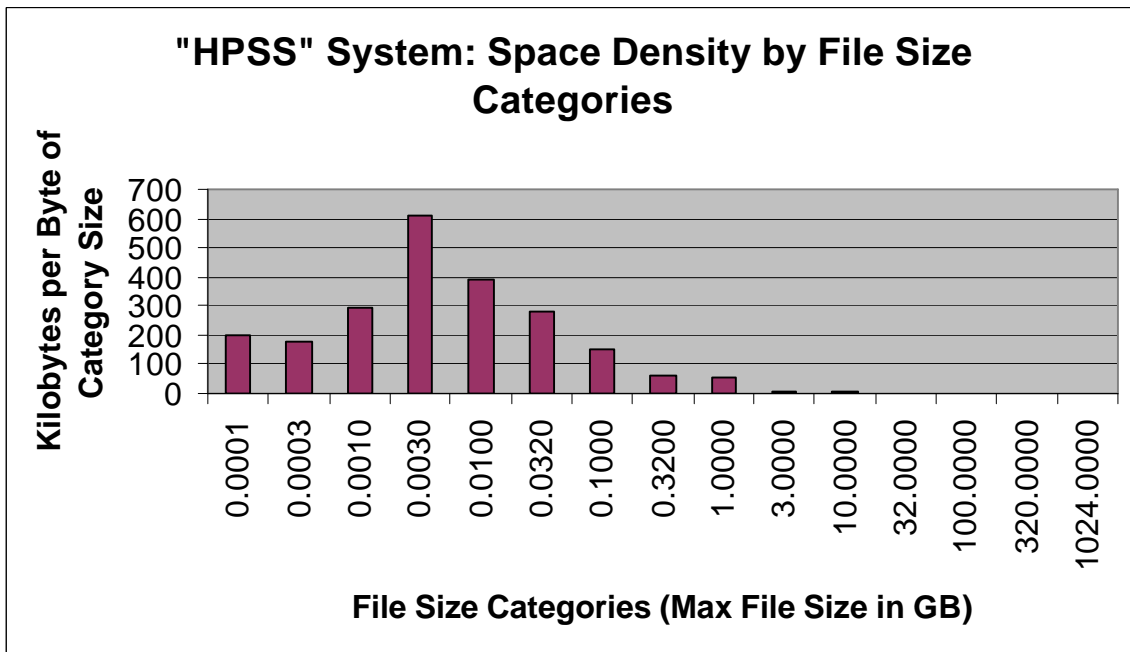


Figure 6. Space Density by File Size Categories for the "HPSS" System

File Access Intervals

A small extract of the access interval distribution is included below. The categories which are not shown have similar behavior – they all show the highest number of accesses on the same day as the previous access, with the rates declining steadily after

that. The data for all files, and for aggregations into small, medium and large file sizes, is shown below for Archive. For "HPSS", we present only the graph of the access intervals for all files combined.

Note that there is some data censoring that we have not made any adjustments for. This data censoring reduces the frequencies of long access intervals and occurs because only the tail end of the analysis period can record long access intervals. For example, for a log file coverage of 600 days, only the last 300 days of the log can result in access intervals greater than 300 days, so the count of 300 day access intervals will be reduced by up to half from the rate that would have been observed with an infinite length log file coverage.

```
#=====
# Archive
# File access intervals by categories of file sizes
#
# Left column: days since last access (original creation not included)
# Subsequent columns: number of files accessed after
#   the given number of days.
#
# File size categories are given in the headers at the top
#=====
#frGB    0.0000    0.0001    0.0003    0.0010    0.0030    0.0100
#toGB    0.0001    0.0003    0.0010    0.0030    0.0100    0.0320
  0    309893    15388    74556    45502    13907    16893
  1      7466     2113     2265    26086     2578     3991
  2      3761     2730      710    10728     1951     2669
  3      2846     7018     349     8556     3433     2215
  4      2140     4732     471      973     1287     1306
  5      2061      245     478     1937      996     1532
  6      1797      97      271     1232      954     1611
  7      2742     416     513     2574      674     1221
  8      1256     186     472     1407      546      739
  9       842     107      76     6302      700      954
 10       796     166     151     1540      598      548
 11      1226     187     222     1137      612     1008
 12      2677     578     376      589     1115      769
 13       433     107     187      476      337      253
 14       318     895     513      272      423      354
 15       577     170      76     1181      475      507
 16       831     1080      30      334      743      515
 17       273      65     388      232      286      224
 18      1599     419     116      244      253      806
 19      1267     826     165      291      191      205
 20      1205     498     261      650      382      699
```

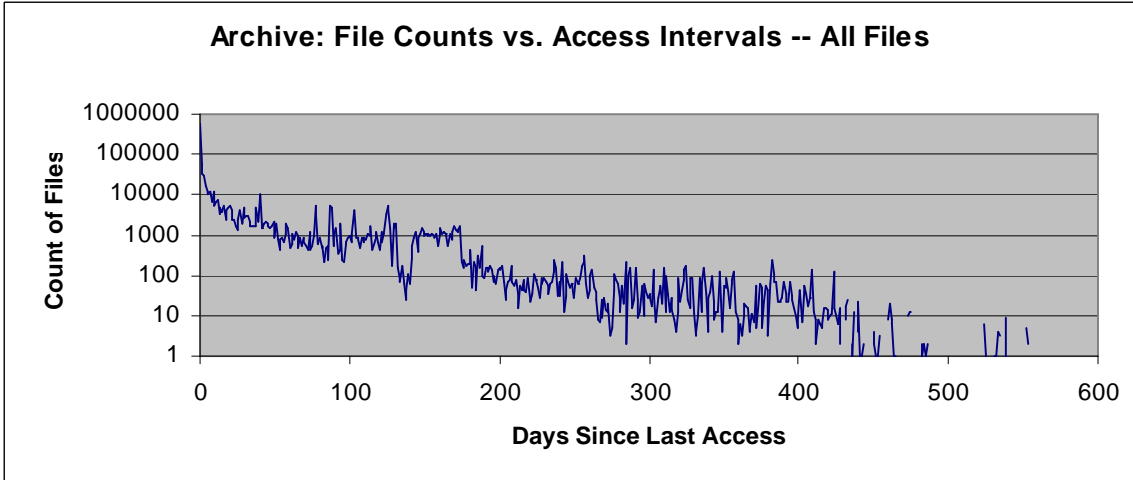


Figure 7. File Counts vs. Access Intervals for the Archive System – All Files

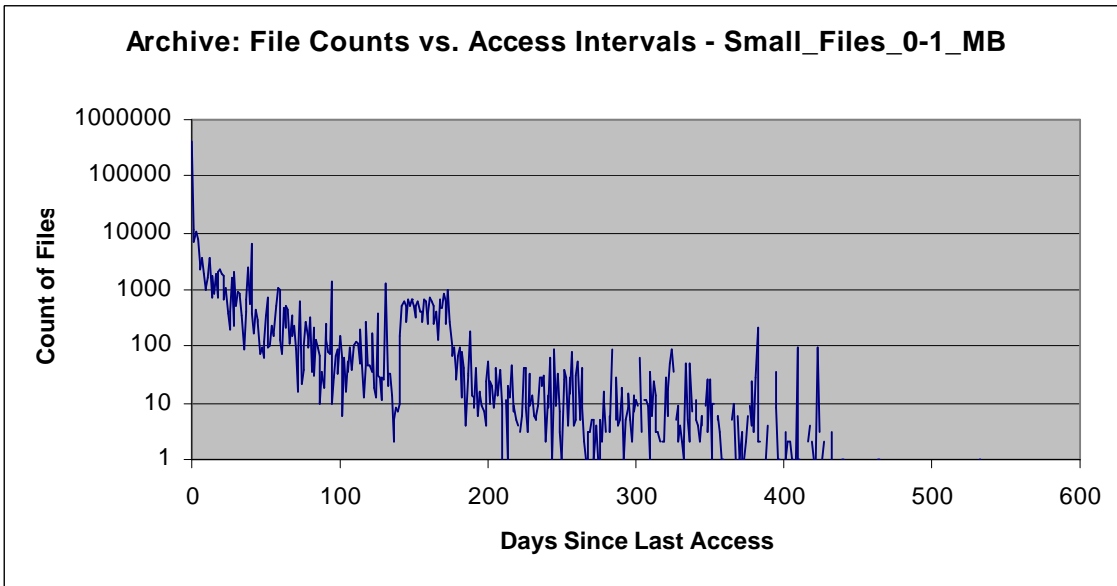


Figure 8. File Counts vs. Access Intervals for the Archive System – Small Files

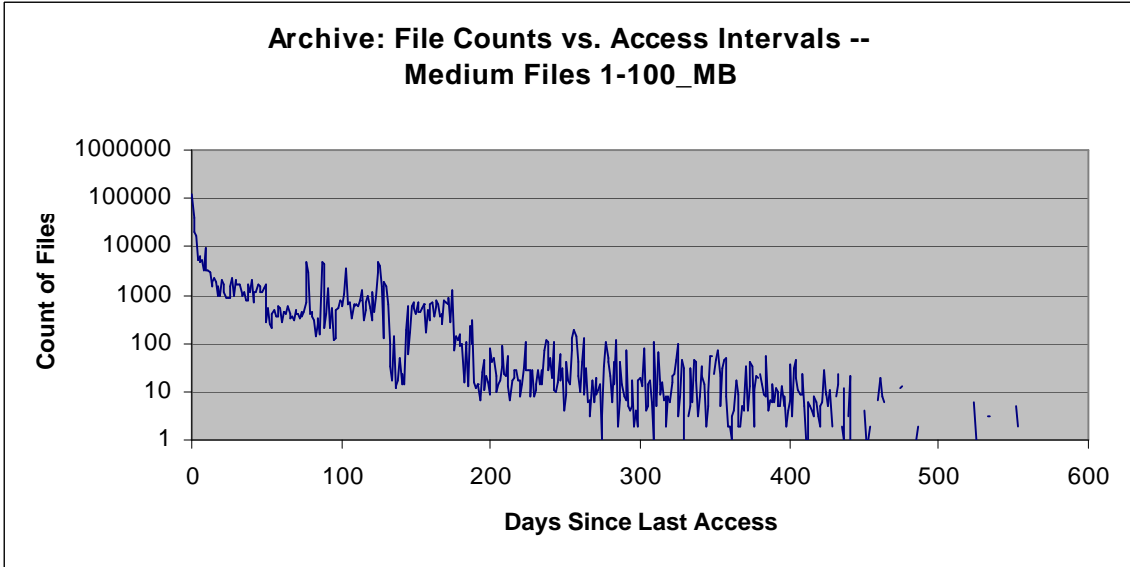


Figure 9. File Counts vs. Access Intervals for the Archive System – Medium Files

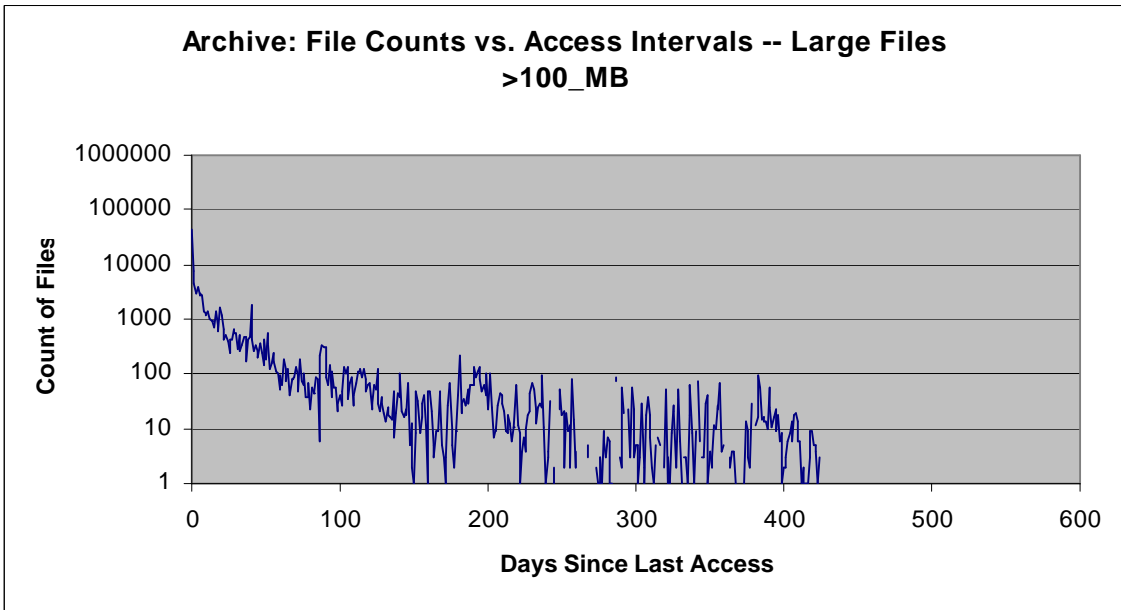


Figure 10. File Counts vs. Access Intervals for the Archive System – Large Files

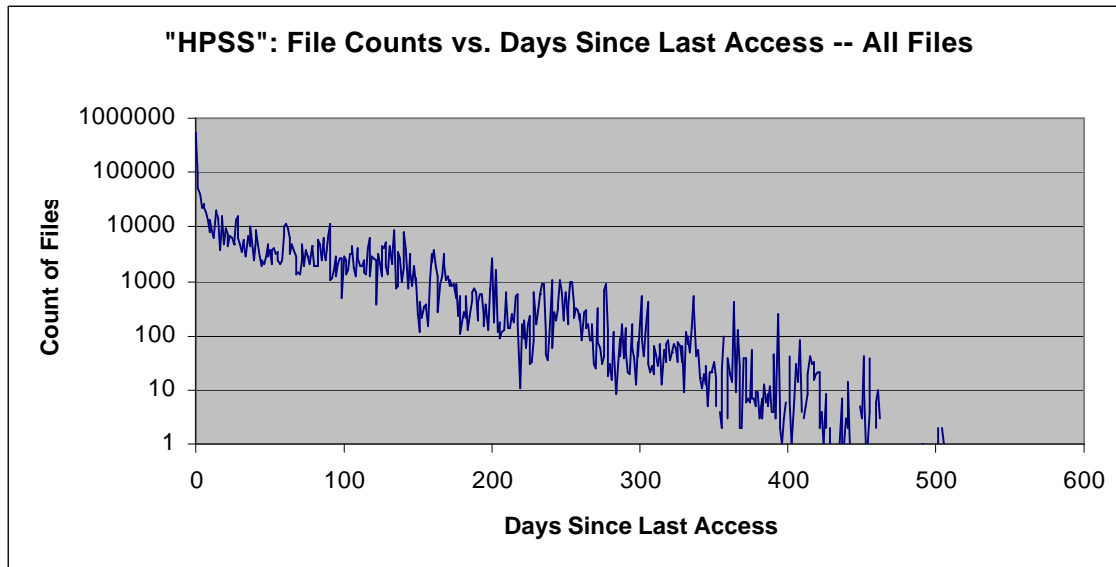


Figure 11. File Counts vs. Access Intervals for the "HPSS" System – All Files

Cache Simulation

To simulate the cache, a PERL script was written which simulates an LRU cache, with parameters for upper and lower bounds on file sizes, the cache size, and (for debugging) a limit on the number of records processed. This script will simulate 2 million operations in 5 to 10 minutes depending on the size of the cache (larger caches take longer). The simulations were run on an Intel laptop with 256 MB of real memory and a 750 Mhz CPU. The script records the number of files and the amount of data

- written into the cache (and thence written to tape),
- read from the cache, and
- read from tape (reads not in the cache).

Of course, the amount of data written is independent of cache size and is already known from the earlier file counts and data summaries. This figure serves as a convenient check. Another check is that the sum of data read from the cache plus data read from tape is a constant which does not depend on the size of the cache.

Statistics Collected

The cache statistics which are collected are fairly simple:

- the number of files which are read from the cache,
- the minimum and maximum cache residence times for all the files being simulated,
- an average cache residence time, computed using two different methods:
 1. an actual average (later versions of the simulator only), and
 2. from the data flow: cache size divided by the amount of data being put into the cache (writes to the cache plus reads from tape into the cache).

A driver script was developed which ran the simulator with a systematic set of parameters to explore the parameter space of Class Of Service and cache configurations. To explore issues related to choices of file size categories, category boundaries were set up at 100 KB, 1 MB, 10 MB, 100 MB, 1 GB, 10 GB, 100 GB, and 1000 GB. All possible combinations of category boundaries were simulated, e.g., 0 to 100 KB, then 0 to 1 MB, and so forth. For each file size category, a simulation was run for cache sizes of 10 GB, 100 GB, 1000 GB, and 10 TB. A sampling of this data is shown below. All file or data sizes are shown in GB. The headings are:

cos_min lower limit of the file sizes simulated (GB)
cos_max upper limit of the file sizes simulated (GB)
cache_sz size of cache being simulated (GB)
tr_cnt count of files simulated (meeting the size limits)
tp_w_dt amount of data written to tape (GB)
tp_r_ct count of tape reads required for data retrieval
tp_r_dt amount of data read from tape (GB)
ch_r_ct count of files read from the cache
ch_r_dt amount of data read from the cache (GB)
ch_rs_m minimum cache residence time during the simulation (days) (1000000
 or blank indicates the cache never filled up during the simulation)
ch_rs_x maximum cache residence time during the simulation (days) (0 or
 blank indicates the cache never filled up during the simulation)
ch_rs_av average cache residence time (days), computed from data flows

Selected "HPSS" cache simulation statistics:

| cos_min | cos_max | cach_sz | tr_cnt | tp_w_dt | tp_r_ct | tp_r_dt | ch_r_ct | ch_r_dt | ch_rs_m | ch_rs_x | ch_rs_av |
|---------|---------|---------|---------|----------|---------|---------|---------|---------|---------|---------|----------|
| 0 | 0.001 | 10 | 1657838 | 195.42 | 161413 | 30.07 | 290277 | 38.66 | 2 | 63 | 24.97 |
| 0 | 0.001 | 100 | 1657838 | 195.42 | 97101 | 19.58 | 354589 | 49.15 | 297 | 377 | 261.86 |
| 0 | 0.001 | 1000 | 1657838 | 195.42 | 94968 | 19.06 | 356722 | 49.68 | 1000000 | 0 | 2625.06 |
| 0 | 0.001 | 10000 | 1657838 | 195.42 | 94968 | 19.06 | 356722 | 49.68 | 1000000 | 0 | 26250.57 |
| 0.001 | 0.1 | 10 | 1830386 | 13766.05 | 872879 | 5973.8 | 50319 | 498.98 | 0 | 9 | 0.29 |
| 0.001 | 0.1 | 100 | 1830386 | 13766.05 | 778081 | 5294 | 145117 | 1178.82 | 0 | 14 | 2.95 |
| 0.001 | 0.1 | 1000 | 1830386 | 13766.05 | 573106 | 3808.8 | 350092 | 2664.02 | 17 | 82 | 32.03 |
| 0.001 | 0.1 | 10000 | 1830386 | 13766.05 | 249289 | 1797.7 | 673909 | 4675.1 | 303 | 440 | 361.74 |
| 0.1 | 1000 | 1000 | 167922 | 105700.1 | 23167 | 10810 | 14815 | 10968.7 | 0 | 20 | 4.83 |
| 0.1 | 1000 | 10000 | 167922 | 105700.1 | 12974 | 4937.6 | 25008 | 16841.4 | 31 | 122 | 50.89 |

Cache Lifetimes

An illustrative way to view the data is to look at cache lifetimes as a function of data flow through the system. "Data flow" is the sum of data flowing into the cache from user writes to the storage system plus data read from tape (into the cache and then to the user). Note that both axes of these curves are logarithmic to cover the wide span of data values. The graphs generally show rather linear behavior over a wide range of values.

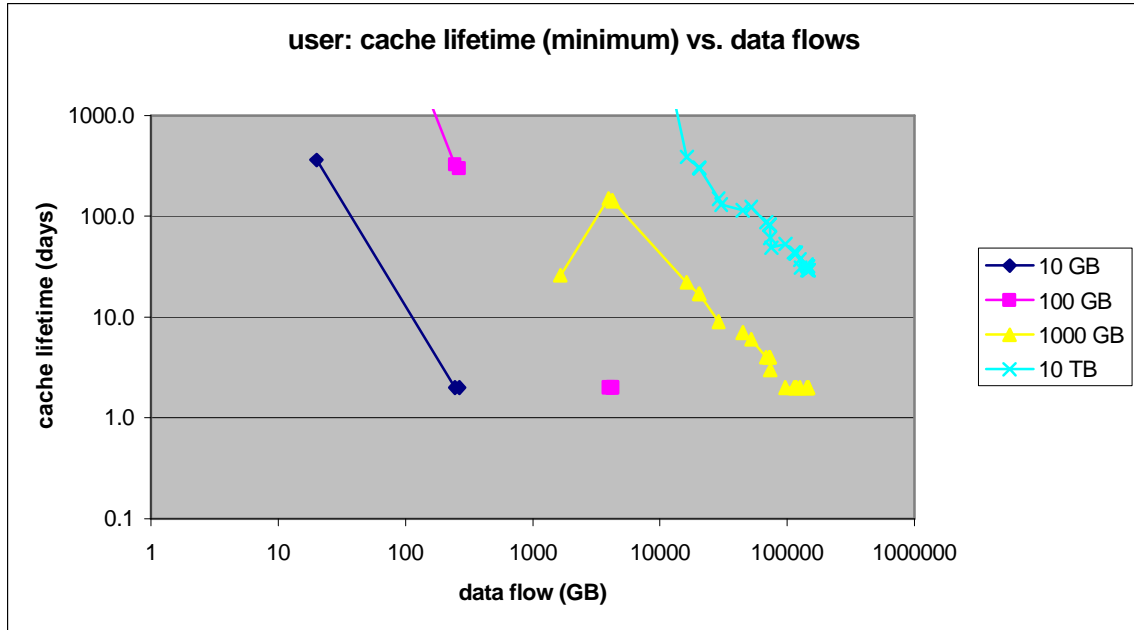


Figure 12. Cache Lifetime vs. Data Flows for the "HPSS" System – Minimum Lifetimes

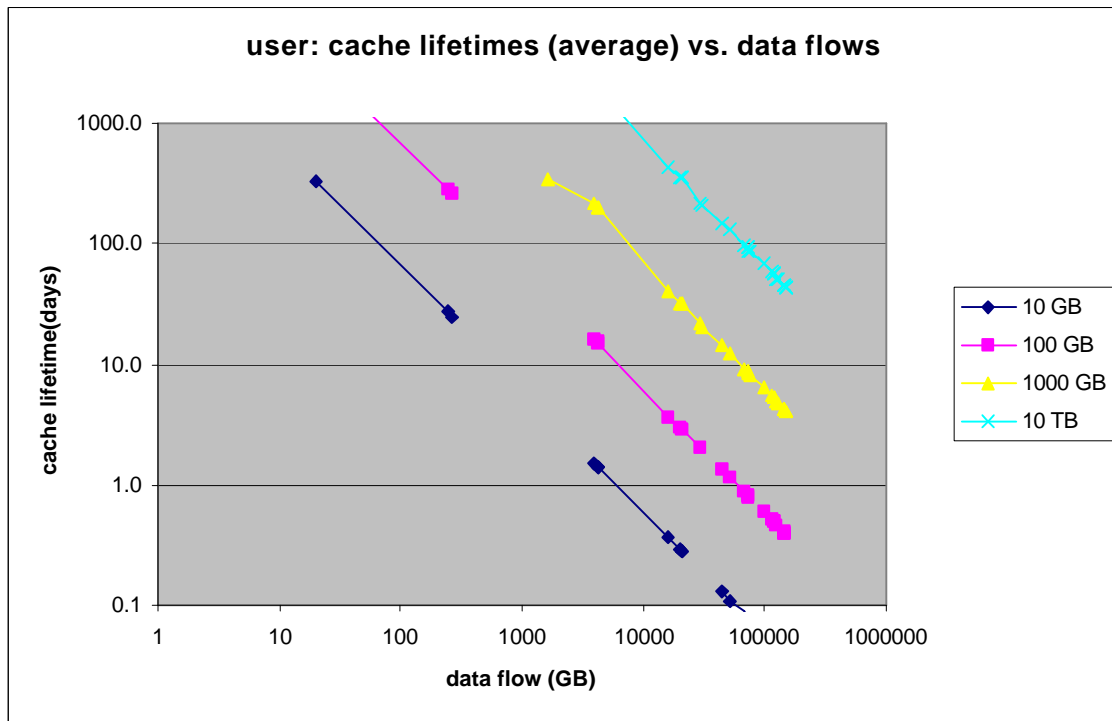


Figure 13. Cache Lifetime vs. Data Flows for the "HPSS" System – Average Lifetimes

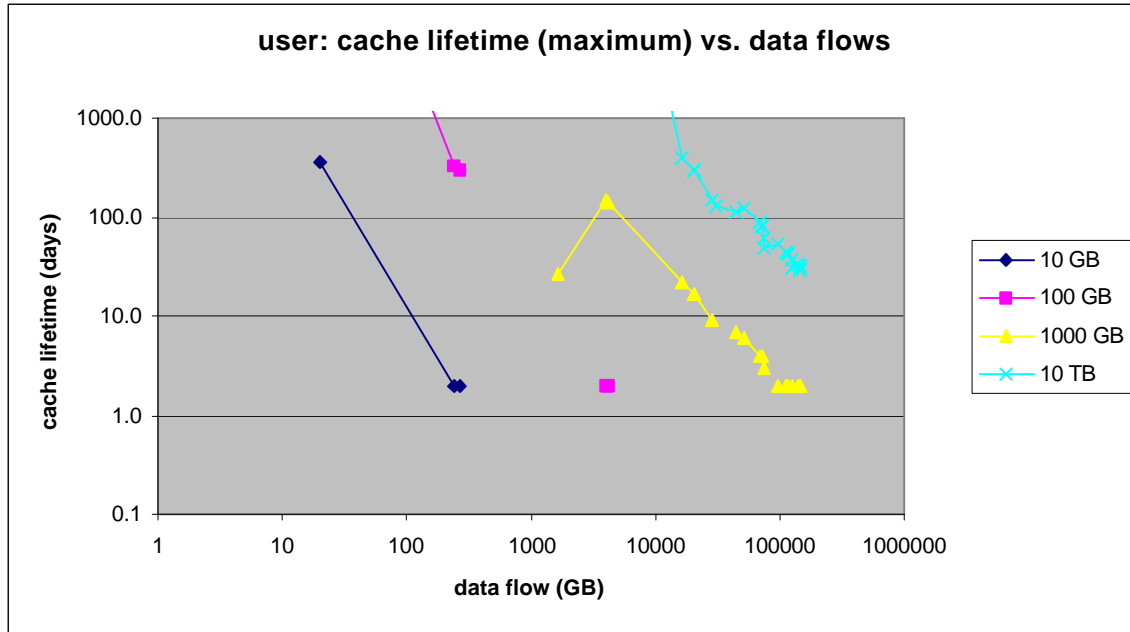


Figure 14. Cache Lifetime vs. Data Flows for the "HPSS" System – Maximum Lifetimes

Sensitivity Analysis: Tape Mounts Required vs. Cache Increments

We also ran the simulation for a few parameters with the size of the cache increased by 1% to see how this would affect the number of tape mounts required for reading data. Also shown is the increase to the next cache size in the standard set of parameters.

| File sizes | Cache increase | Tape mount decrease | Mount decrease/GB |
|----------------|------------------|---------------------|-------------------|
| 0 to 1 MB | 10 to 10.1 GB | 169 | 1690 |
| 0 to 1 MB | 10 to 100 GB | 64312 | 715 |
| 1 to 100 MB | 100 to 101 GB | 330 | 330 |
| 1 to 100 MB | 100 to 1000 GB | 204975 | 228 |
| 100 MB to 1 TB | 1000 to 1010 GB | 1218 | 121.8 |
| 100 MB to 1 TB | 1000 to 10000 GB | 10193 | 1.1 |

Sensitivity Analysis: Disk Allocation Size

One of the configuration parameters in HPSS is the choice of the disk allocation size. To study the impact of this choice, simulation runs were made with several choice of disk allocation size. These runs were all made for a class of service from 0 to 1 MB with 100 GB of cache. As can be seen from the graph, the number of tape reads required to satisfy the read requests is relatively constant up through 0.25 MB, then shows an increase by about 50% as the allocation size is increased to 1 MB. The number of requests satisfied from the cache shows the same decrease at 1 MB, but the change is smaller in proportion. The minimum cache lifetime is more interesting, showing a decrease starting about 16 KB and showing a steady drop from there all the way to 1 MB. All these results are consistent with the file inter reference intervals described earlier; since the number of references is concentrated in the short inter reference intervals, the cache effectiveness

remains high even though the cache lifetime is declining. In the graph below, "tp_r_cnt" is the count of tape reads required to satisfy the read requests, "ch_r_ct" is the count of cache read requests required to satisfy the read requests, and "ch_rs_m" is the minimum cache residency time observed during the simulation using the specified disk allocation size.

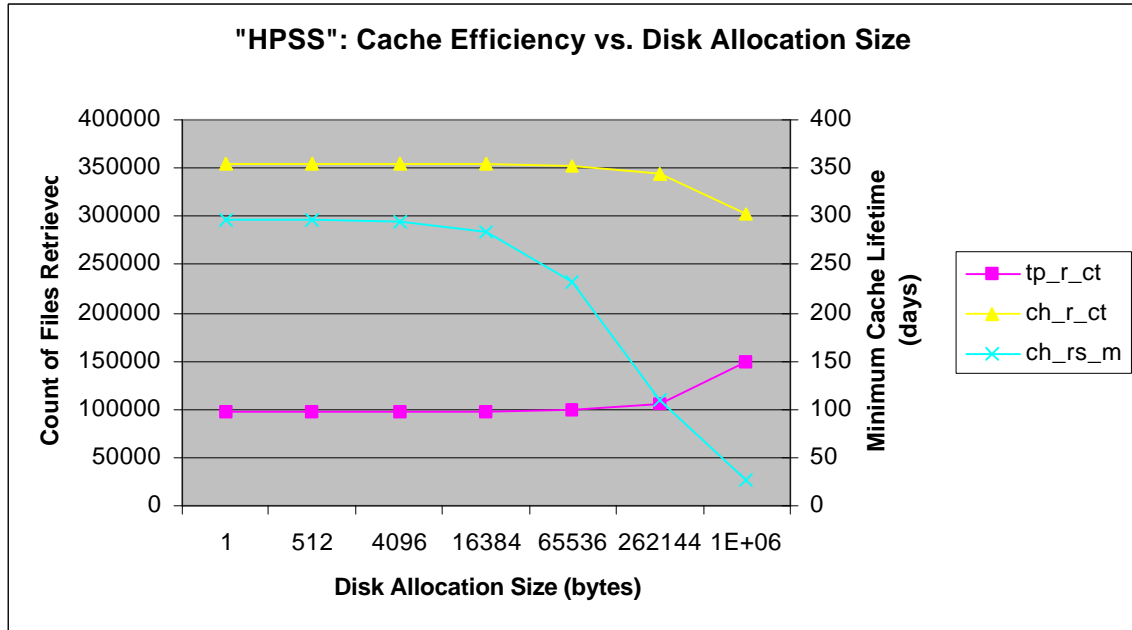


Figure 15. Cache Efficiency vs. Disk Allocation Size for the "HPSS" System

Sensitivity Analysis: Purge Policy Choices

Classes of Service use "hierarchies" to manage their storage, and hierarchies are composed of one or more "storage classes." Storage classes are more or less uniform types of devices, either disk or tape. Data movement between the storage classes is done according to "migration" and "purge" policies. Our main concern here is with purge policies, as these control the removal of files from the disk cache. HPSS in version 4.1 provides 3 choices of purge policy:

- Purge Record Creation Time (P) – this provides First In First Out behavior, suitable to meet certain fairness requirements,
- File Creation Time (F) – to meet certain fairness requirements, and
- Last Data Access Time (L).

We made a few runs to compare the Purge Record Creation Time behavior with the Last Data Access Time behavior. It was surprising how little difference it made:

| File sizes | Cache | Tape mounts(P) | Tape mounts(L) | Mount decrease |
|----------------|----------|----------------|----------------|----------------|
| 0 to 1 MB | 10 GB | 162169 | 161424 | 745 (0.5%) |
| 0 to 1 MB | 100 GB | 98590 | 97111 | 1479 (1.5%) |
| 1 to 100 MB | 100 GB | 779417 | 778180 | 1237 (0.2%) |
| 1 to 100 MB | 1000 GB | 579470 | 573159 | 6311 (1.1%) |
| 100 MB to 1 TB | 1000 GB | 23613 | 23188 | 425 (1.8%) |
| 100 MB to 1 TB | 10000 GB | 13413 | 12992 | 421 (3.2%) |

Observations

Our observations from this simulation work is that our HPSS systems are rather dominated by the incoming write traffic and this tends to wash out nuances of behavior that might otherwise be observed. In particular, the write traffic tends to flush the cache on a very regular bases, producing two main effects:

1. The cache will never achieve near 100% coverage so the finer details of cache operation are mainly irrelevant, and
2. The constant flow of data through the cache makes it's operation rather linear and fairly easy to predict. This is particularly true for cache residence times.

We were surprised by the lack of affects in some of our sensitivity analyses. The sensitivities to disk allocation sizes and to the purge algorithm choice were less than we had anticipated.

On the other hand, the sensitivity of tape mount requests to cache size could provide a convenient mechanism to optimize the allocation of disk resources to caches for the various classes of service.

Recommendations

Several problems face the staff who are designing and installing HPSS. It is difficult to estimate the workload until the system has been in operation for some time, as the very existence of the system is likely to cause workload patterns to change from what they were when estimates of system load were initially made.

Most system administrators will be concerned with choosing appropriate classes of service and allocating disk resources among the classes of service to provide the best service to the users. The type of simulation technique described here can be used to explore options, provided *the workload is similar in character*. Our results suggest that the administrators need not worry too much about disk allocation sizes or purge policy choices. This type of simulation could also be used to make economic tradeoffs between disk and tape hardware, but we feel that a performance based approach (to provide users with good service) is more useful than an economic approach, and we have not applied such an economic approach to our own systems.

The file access interval data from the workload analysis is probably representative of a variety of systems and could be used to generate a synthetic workload for simulation runs

such as we have done with real data. This would allow sites to explore options before installation, when insights are most needed and useful.

Future Work

An easy extension of this work is to send the results of the simulation runs into an optimizer which would generate optimal disk cache sizes for a specific choice of file sizes in the Classes of Service. A further extension would undertake the automatic selection of file size boundaries for classes of service.

To accomplish these optimizations, it would be useful to have analytic expressions which would characterize the relevant parameters across the needed range of input parameter spaces.

Caching strategies which explicitly account for the heavy write traffic should be reviewed.

Conclusions

Our work has uncovered few surprises, but it has served to confirm our expectations. In the areas of purge policy choices and disk allocation sizes, we were somewhat surprised by the lack of sensitivity to these choices.

We have definitely gained a better insight into effective cache allocation and utilization.

The importance of write traffic on cache performance suggests that more work on cache algorithms that explicitly recognize write traffic might be well rewarded.

Acknowledgement

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

1. <http://www.sdsc.edu/hpss/>
2. <http://hpcf.nersc.gov/storage/hpss/>
3. R. A. Coyne and H. Hulen, "An Introduction to the Mass Storage Reference Model, Version 5," *Proc. 12th IEEE Symp. Mass Storage*, Monterey, CA, IEEE Computer Society Press, April 1993.
4. IEEE Storage Systems Standards Working Group public documents, http://www.ssswg.org/public_documents/MSSRM/V5toc.html
5. <http://hpcf.nersc.gov/storage/hpss/stats/monthly.html>

Disclaimer

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, or The Regents of the University of California.

Ernest Orlando Lawrence Berkeley National Laboratory is an equal opportunity employer.